International Journal of Electrical and Electronics Engineers
Vol. No. 9, Issue No. 01, January-June 2017
ISSN (O) 2321-2055
ISSN (P) 2321-2045

STEGANOGRAPHY USING 2LSB PARITY CHECK AND BPCS TECHNIQUES

Prof . Dipti Dighe[1] , Swati Dubey[2], Prof. Shubhangi Takawane[3]

[1,2,3] *Electronics and telecommunication Engineering Department,*

G.S. Moze College of Engineering, Pune, (India)

ABSTRACT

Steganography means covered writing and is the art of hiding secrete message within another innocuous message or carrier. The carrier could be any medium used to convey information. There are various tecghniques are available to hide information within the carrier. This paper includes 2LSB parity check and Bit plane complexity segmentation (BPCS) steganography techniques to hide data inside an image files which are simple and very effective.

Keywords: Steaganography, 2LSB Parity Check, Bit Plane Complexity Segmentation.

I. INTRODUCTION

In many situations, transmitting a cryptographic message draws unwanted attention. The use of cryptographic technology may be restricted or forbidden by law. In the computer world, it is very important to keep secret information secret, private information private, and when profits are involved, protect the copyrights of data. To accomplish these difficult tasks, new methods based on the principle of steganography is being developed and used. Steganography is the art and science of communicating in a way which hides the existence of the communication [1]. Data embedding or data hiding is a technique that enables us to secretly embed extra data into a file such as an image file, a movie file, and an audio file. The embedded data disappear into the file and we cannot recognize the embedded data in the file. The word steganography comes from the Greek 'steganos', which means covered or secrete and 'graphy' means writing or drawing [3]. Therefore, steganography means covered writing.

II. TECHNIQUES

There are many different techniques are present for steganography. Here we are focusing on two techniques which are Bit Plane Complexity Segmentation and other one is 2 LSB steganography .

2.1 Parity Check 2LSB Technique

This technique is based on RGB images. The two least significant bits of the red channel will be used as an indication to the existence of hidden data in green and blue channels. Before embedding the data bits, the parity of the data has been checked [4][5]. RGB image consist of 3 colors red, green & blue. Image component is given by equation (2.1). Here R(x,y) is used as a controlling element & pair of data bits are embedded into G(x,y) & B(x,y). For any sequence of message bit pairs (m1, m2) (m3, m4). .. .. (m i-1,mi) ,we will compare the message bits with current G(x,y) and B(x,y) and by using following tables data bits will get added into pixels.

International Journal of Electrical and Electronics Engineers
Vol. No. 9, Issue No. 01, January-June 2017
ISSN (O) 2321-2055
ISSN (P) 2321-2045

Whether to embed odd or even parity data is decided by performing modulus operation on R(x,y) which is obtained by equation (2.2).

$F(x,y)=R(x,y)+ G(x,y)+ B(x,y)$ …(2.1)

$(R(x,y) + 2) \bmod(4) = 0$ …(2.2)

If equation (2.2) satisfies the condition, difference between message bit pairs & two least significant bits of G(x,y) is calculated using equation (2.3). If OE is less than ±2, even parity data is embedded into G(x,y).

$OE = G(b1,b0) -(mi-1,m i)$ … (2.3)

Where, b0 and b1 are two least significant bits of pixel. Same technique is used for embedding in B(x,y). OE decides whether to embed or not. If equation (2.2) does not satisfy the condition, then pair of odd data bits is embedded. OE is used to control embedding data rate. The OE variable affects probability of embedding payload. For maximum efficiency, value of OE must be in between -1 to +1. Table 2.1 shows the embedding scheme for message bits if two least significant bits of red color is divisible by 2 then and only then embed the even parity bits of the message. The following steps are used to embed data in pixels:

2 LSBs of red	2 LSBs of green	2 LSBs of blue
00	Add even parity data	Add even parity data
01	Add odd parity data	Add odd parity data
10	Add even parity data	Add even parity data
11	Add odd parity data	Add odd parity data

Table. 2.1. Embedding scheme

Otherwise place the odd parity bits in to the green color and blue color component of that particular pixel according to the following steps

If data bits are 00 then

$$\mathbf{G(x,y)} = \begin{cases} \mathbf{G(x,y)} & \mathbf{for\ g00,g10,g11} \\ \mathbf{G(x,y) - 1} & \mathbf{for\ g01} \end{cases} \quad \ldots(2.4)$$

If data bits are 11then

$$\mathbf{G(x,y)} = \begin{cases} \mathbf{G(x,y)} & \mathbf{for\ g00,g01,g11} \\ \mathbf{G(x,y) + 1} & \mathbf{for\ g10} \\ \mathbf{For\ R(x,y)\ mod(\ 2\ ) \ != 0} & \end{cases} \quad \ldots(2.5)$$ … (2.6)

If data bits are 01 then

International Journal of Electrical and Electronics Engineers
Vol. No. 9, Issue No. 01, January-June 2017
ISSN (O) 2321-2055
ISSN (P) 2321-2045

$$G(x,y) = \begin{cases} G(x,y) & \text{for } g01, g11 \\ G(x,y) - 1 & \text{for } g10 \\ G(x,y) + 1 & \text{for } g00 \end{cases} \quad \ldots(2.7)$$

If data bits are 10 then

$$G(x,y) = \begin{cases} G(x,y) & \text{for } g10, g00 \\ G(x,y) - 1 & \text{for } g11 \\ G(x,y) + 1 & \text{for } g01 \end{cases} \quad \ldots(2.8)$$

Equation (2.4), (2.5), (2.7) and (2.8) are used to embed the message bits in to green color. Same equations can be used to embedding in blue color. We have placed the message bits in B(x,y) after performing the operations on G(x,y).

If the contents of R=00, OR 10 and at the same time data bit s are 00 or 11

	d00	d11
For G=00	G=G	DON'T ADD
For G=01	G=G-1	DON'T ADD
For G=10	DON'T ADD	G=G+1
For G=11	DON'T ADD	G=G

Table 2.2 Embedding Scheme when R = 00 OR 10

If the contents of R=01, OR 11 and at the same time data bit s are 00 or 10

	d01	d10
For G=00	G=G + 1	DON'T ADD
For G=01	G=G	G=G+1
For G=10	G=G -1	G=G
For G=11	DON'T ADD	G=G - 1

Table 2.3 Embedding Scheme when R = 01 OR 11

Worst condition: If two lsb's of red color is 00 or 10, and at the same time if data bits which we have to embed having odd parity, then contents of R(x,y) is changed by using equation

$$R(x,y) = R(x,y) - 1 \quad \ldots(2.9)$$

International Journal of Electrical and Electronics Engineers
Vol. No. 9, Issue No. 01, January-June 2017
ISSN (O) 2321-2055
ISSN (P) 2321-2045

Same problem may occur if equation (2.6) satisfies the condition & data bits are having even parity. It can be solved by using equation (2.9). Here pixel value will be changed without embedding data bits. This is undesired operation which helps to degrade the image. But embedding capacity is increases. Figure shows the condition when data is not embedding in to the pixel. It will decrease the embedding rate of the system. But this is useful for improving the security. Since some of the pixel doesn't carry any information, it is difficult to detect the pixel which carries the information.

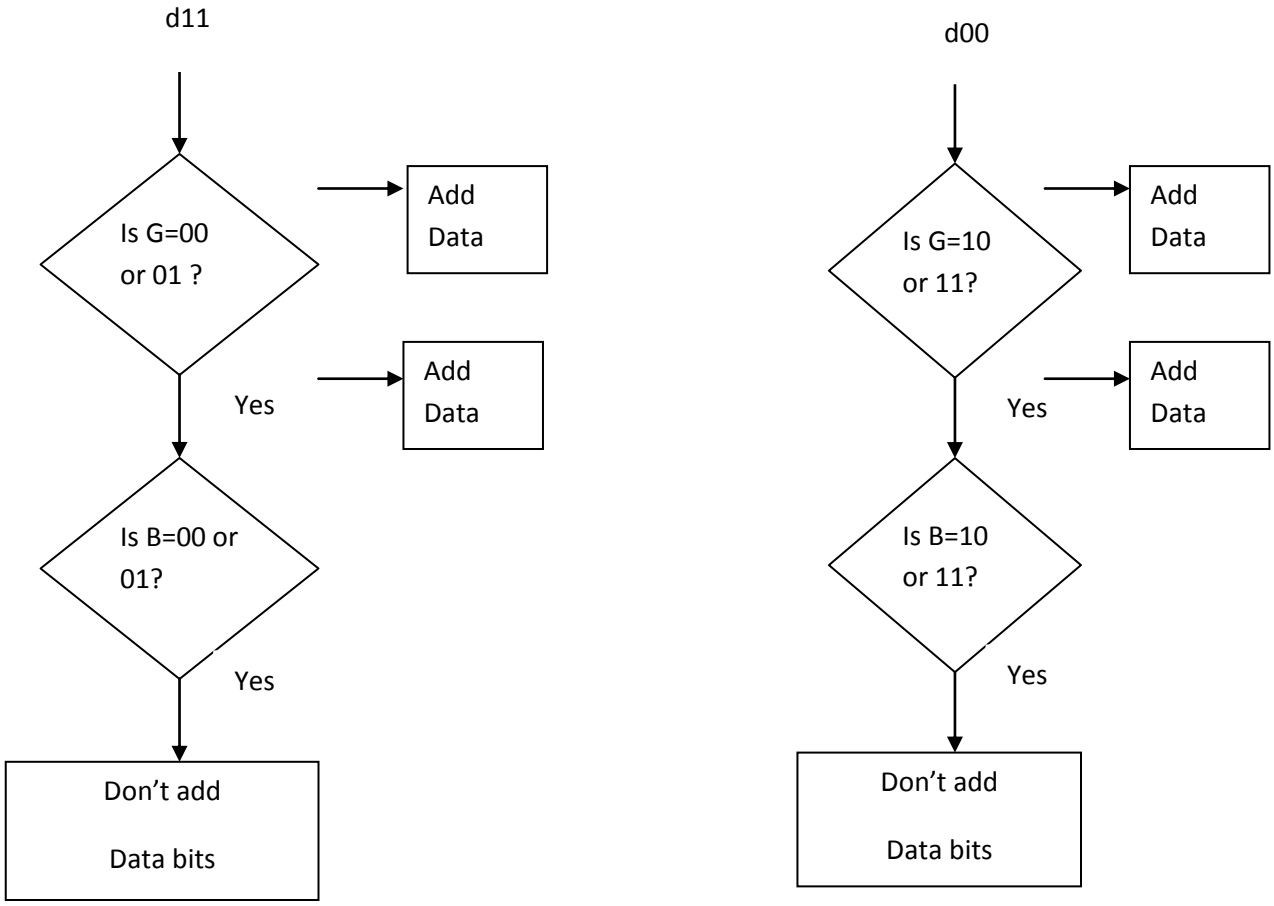

Fig. 2.1 Worst case chart

2.2 Bit Plane Complexity Segmentation Technique

In BPCS, the vessel image is divided into "informative region" and "noise-like region" and the secret data is hidden in noise blocks of vessel image without degrading image quality. We can use this property for our information hiding (embedding) strategy[7].

2.2.1Complexity Measure

The first step in BPCS Steganography is to find "complex" regions in the image where data can be hidden imperceptibly. There is no universal definition for the complexity of an image (or a region of an image).

a) Complexity measure based on length of black and white border (α)

This measure is defined on the 4-connected neighbourhood of a pixel. The total length of the black-and-white border is defined as the sum of the colour changes along the rows and columns in the image. For example, a single white pixel surrounded by 4 black pixels, i.e. ,having all its 4-connected neighbours as black pixels, will

International Journal of Electrical and Electronics Engineers
Vol. No. 9, Issue No. 01, January-June 2017
ISSN (O) 2321-2055
ISSN (P) 2321-2045

have a border length of 4 (2 color changes each along the rows and columns). Extrapolating this idea to a square binary image of size $2 * 2^N * (2^N - 1)$, for the black and white checker board pattern $(2^N - 1)$ Change along each of the 2^N rows plus the same along the columns. The image complexity measure, α, is then defined as the normalized value of the total length of the black and white border in the image [2.2], i.e.

$$\alpha = \frac{k}{2X2^N X(2^N - 1)}, \quad 0 \leq k \leq (2X2^N X(2^N - 1)$$

Here 'k' is the actual length of the black and white border in the image. It is evident that α lies within [0, 1] .

b) Complexity measure based on the number of connected areas (β)-

This measure is again based on the 4-connected neighbourhood. 'β' is defined as

$$\beta = \frac{m}{2^N X 2^N}$$

Here 'm' is the number of connected areas in the 2^N X 2^N square binary image. It is easily seen that β lies in $[1/(2^N \text{ X } 2^N)]$,1] with the maximum in the range obtained for the checker board pattern and the minimum obtained for the plain white or plain black image.

The assumption that the image is a square of size 2^N X 2^N severely cripples the applicability of these measures to all images, considering that images are not always perfectly square. To make these complexity measures more generic, they are applied to each exclusive 2^n x 2^n block, where n is typically between 2 and 4, of any M x N image. The only condition is that M and N have to be divisible by 2^n. This limits higher values of n. Very small values for n (n=1, 2) provide too much spatial localization for the complexity measures to be meaningful. In practice, n is fixed at 3 so that the complexity measure is applied to each exclusive 8x8 block of the image.

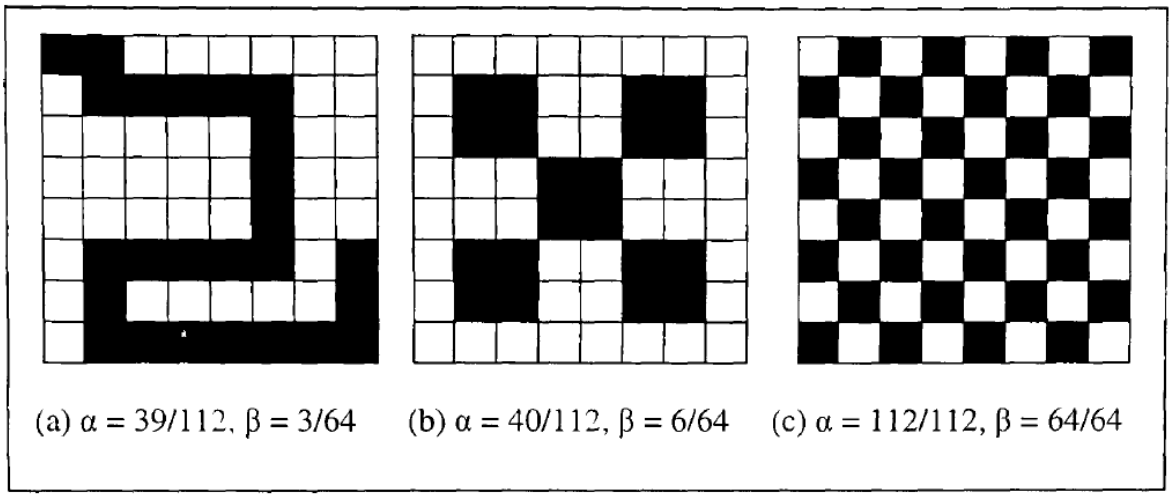

Fig. 2.2 α and β values for some 8x8 blocks

Above figure shows α and β values for 2 typical 8x8 blocks. In practice is it found that, for 8x8 blocks, the 'α' measure is more or less uniformly distributed in [0, 1] while the β measure tends to have a definite peak at β=0.2. Hence the 'α' measure is preferred for the BPCS application.

2.2.2 Bit Plane Slicing Concept in BPCS

The operation of splitting the image into its constituent binary planes is called Bit-Plane Slicing. Each 8-bit gray-scale image can be split into 8 binary planes, one plane for each of the 8 significant bits in the 8-bit binary

International Journal of Electrical and Electronics Engineers
Vol. No. 9, Issue No. 01, January-June 2017
ISSN (O) 2321-2055
ISSN (P) 2321-2045

representation of image intensity values. 24-bit color images are composed of three 8-bit planes, one each for Red, Green, Blue and can be split into 24 binary planes.

The bit plane slicing can be better understood with the help of figure 2.2. In an 8-bit image, intensity of each pixel is represented by 8-bits. The 8-bit image is composed of eight 1-bit plane regions from bit plane '0' (LSB) to bit-plane '7' (MSB). Plane '0' contains all lowest order bits of all pixels in the image while plane '7' contains all higher order bits. Bit plane Slicing is useful for image compression. Complexity of each bit–plane pattern increases monotically from MSB to LSB[8].

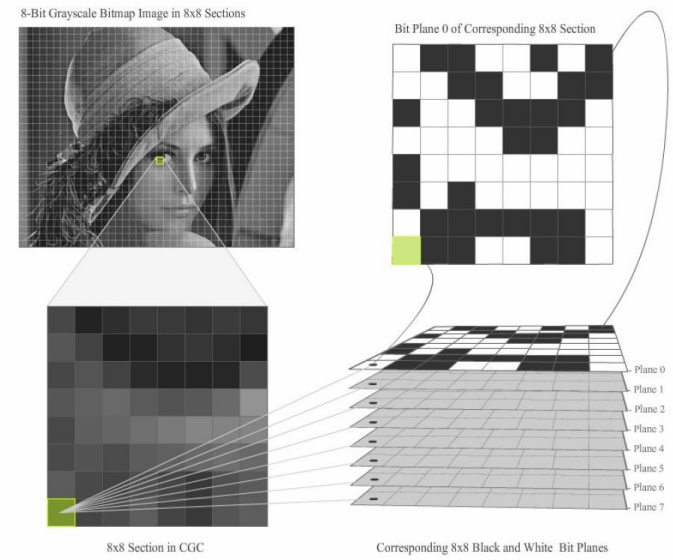

Fig. 2.2 Bit-Plane Slicing

2.2.3 Canonical Gray Coding System

Bit-plane slicing can be done in the Pure-Binary Coding system wherein the intensity values (for each plane in the case of RGB images) are represented as 8 bit binary numbers, but it suffers from a serious drawback. Consider an 8-bit image where a large portion of the image is composed of pixels whose intensity values alternate between 127 and 128. 127 is 01111111 in binary and 128 is 10000000. Thus all 8 corresponding bit-planes for the 2 pixels are different (i.e. distance is 8). This idea of two numbers begin very similar in value yet differing greatly in their binary representation, on a bit by bit basis, is called the 'Hamming Cliff'. In such a region, if these 2 gray levels are sufficiently randomly distributed, all the 8 planes, including the MSB plane, corresponding to these regions would appear complex and hence would be replaced by data to be hidden. After embedding, 01111111 could easily become 11111111 and 100000000 could become 00000000 and what was an intensity difference of just 1 gray level and was rather unnoticeable, now becomes a difference of 256 and appears as an eccentric white pixel next to black pixel or vice-versa. This problem is easily alleviated by using the principle used in some electromechanical applications of digital system where sensors are required to produce digital outputs that represent a mechanical position. The coding system used is called the Canonical Gray Coding System (CGC), where successive decimal numbers differ in their representation by just one bit. It is a canonical system, as the binary system and the gray code system share a one-to-one correspondence. The 2 numbers in the above example, 127 and 128, would be represented in CGC as 01000000 and 11000000, respectively, and hence would not differ by more than 1 bit. Thus, the first step in BPCS steganography is to

International Journal of Electrical and Electronics Engineers
Vol. No. 9, Issue No. 01, January-June 2017
ISSN (O) 2321-2055
ISSN (P) 2321-2045

convert the absolute intensity values into CGC by a 1-to-1, PBC-to-CGC mapping. This is followed by bit-plane decomposition on the CGC values, and the 8 binary images obtained are called the CGC images. The CGC images don't suffer from Hamming Cliff as regions that are rather smooth in the original image result in very few changes in the higher bit planes, and these regions are appropriately determined unsuitable for embedding data.

Since 24-bit RGB color images provide a very high capacity for data hiding applications. BPCS Steganography can also be applied, effectively, to 8-bit grayscale images

2.2.4 Resource Blocks and Conjugation Operation

Once the 24-bit image (base image) has been split into its 24 constituent bit-planes, and the complexity, α, of each exclusive 8x8 block in each of the 24 bit-planes has been found, the complexity of each block is compared with a threshold, α_0.

If the measured complexity is greater than the threshold value (i.e. If $\alpha > \alpha_0$), the block is deemed complex enough to be replaced by data blocks. The standard value used for threshold complexity (α_0) is about 0.3. The data chunks that replace the complex blocks in the bit-plane image are called Data Blocks or Resource Blocks. The resource blocks are chunks of data obtained from any ASCII-encoded file (data or image files that can be read as a string of ASCII character) called the Resource File. The resource file could be a text file or Word document or even an image. Each 8 byte block of a resource file forms an 8x8 resource block with the 8-bit binary representation of each byte forming the row of the 8x8 block. After the resource file broken into 8-byte chunks and cast into 8x8 binary resource blocks, they are ready to replace the complex blocks in the bit-plane images, however, with one problem. A complex block denotes a block that appears noisy, and modifying such a block wouldn't be perceptible, unless, the modification result in making the block less complex than the threshold complexity value, α_0.

Consider the block shown in Figure 2.3 (b). This is a frequently encountered block as word documents contain numerous of blank space. The complexity of this block is 0.1429, which is far less than the α_0 value usually used. If this block replaces a complex block in the bit-plane image, a definite discrepancy arises, especially if it is in one of the higher order bit-planes. Also the decoding module will not recognize the block, as it assumes that only the complex blocks have been replaced and hence only the complex blocks have valid information. To overcome this problem, the 'Conjugation Operation' is introduced.

Figure 2.3 (b) shows the most complex 8x8 block possible, with a complexity of 1. This block is denoted as W_c with its top-left value being 1. A similar checkerboard pattern with complexity 1 can be formed with the top-left value to be a 0, and that is denoted by B_c. The all white and all black blocks are denoted by W and B .W_c is used for all future explanations, although all of it would apply to B_c as well. The W_c block has a special property that when it is XORed (exclusive OR operation) with a non-complex block, P (say), of complexity say $\alpha_n < \alpha_0$, then the resulting block , P*, has a complexity of $(1\ \alpha_n) > \alpha_0$. As with any XOR operation, the block P can be easily retrieved by XORing again with W_c. This operation of changing the complexity of a block by XORing with W_c is called the 'Conjugation Operation' and is denoted by '*'. Figure 2.3(a) shows non-complex (or simple) block, P (say), 2.3(b) is the perfectly complex block, W_c and 2.3 (c) is the conjugated block, P*, obtained by XORing corresponding pixels in Figure 2.3 (a) and (b)

International Journal of Electrical and Electronics Engineers
Vol. No. 9, Issue No. 01, January-June 2017
ISSN (O) 2321-2055
ISSN (P) 2321-2045

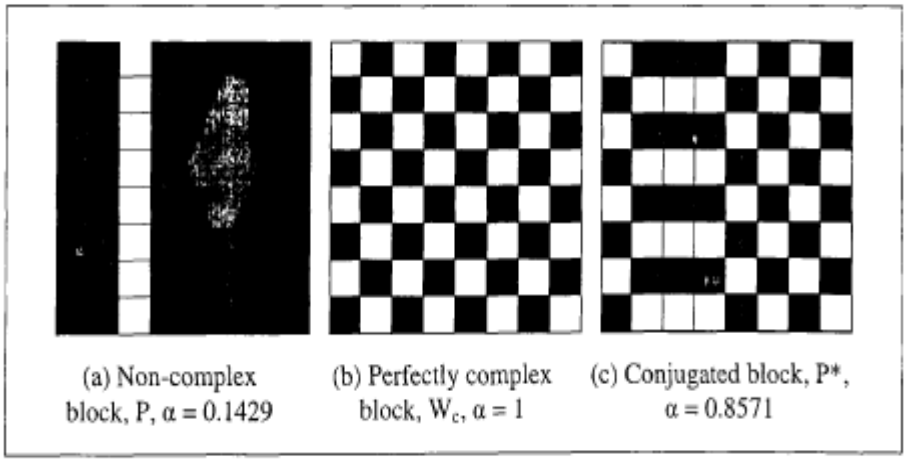

Fig 2.3 Non-complex Block Is Conjugate With Perfectly Conjugate Block

The important properties of the conjugation operator can be summarized as:

1. $(P^*) = 1-\alpha(P)$
2. $(P^*)^* = P$

The first property is used at the encoder to make the non-complex resource blocks complex, while the second property is used at the decoder to retrieve the original block. It happens that all the resource blocks need to be made complex, as most of them are complex on their own. It becomes important to keep track of which blocks have been conjugated. This is done by using a Conjugation Map. For every 8 byte block of the resource file, one bit is appended to the conjugation map to indicate if the block has been conjugate. A '1' implies the resource block was embedded as is. The conjugation map is finally embedded after embedding all the resource blocks[7].

2.7 CONCLUSION

In this paper we have explained 2LSB steganography technique using parity check in which we can hide data into two least significant bit of image depending on parity of secrete data. This is very easy and yet very effective technique. In BPCS we can hide data in images by increasing the complexity of image by doing xor operation between non complex and complex block. In Future we can use cryptography technique for encryption and then embed data using steganography technique to increase the security of information.

REFERENCES

[1] Chen Ming Zhang Ru Niu Xinxin Yang Yixian, "Analysis of Current Steganography Tools: Classifications & Features"‖ Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH- MSP'06) 0-7695-2745-0/06 © 2006 IEEE.

[2] Dipti.G.Dighe,Prof. N.D.Kapale, "Random Insertion Using Data Parity Steganography Technique" International Journal of Engineering Science and Innovative Technology (IJESIT) Volume 2, Issue 2, March 2013.

[3] "High Capacity Data Hiding System Using BPCS Steganography" by Yashvant Shrinivasan.

[4] Rajkumar , Rahul Rishi, Sudhir Batra, "A New Steganography Method for Gray Level Images usingParity Checker", International Journal of Computer Applications (0975 – 8887) Volume 11– No.11,December 2010.

[5] Sampath Kumar Dara, Harshavardhan Awari, "Tree Based Parity Check Scheme for Data Hiding", International Journal Of Computational Engineering Research (ijceronline.com) Vol. 2 Issue. 5, September 2012.

[6] Andrew D. Ker, "Steganalysis of Embedding in Two Least-Significant Bits", IEEE Transactions On Information Forensics And Security, Vol. 2, No. 1, March 2007.

[7] Eiji Kawaguchi and Richard O. Eason "Principle and Applications of BPCS-Steganography" Kyushu Institute of Technology, Kitakyushu, Japan – University of Maine,Orono, Maine.

[8] "BPCS steganography" By Steve Beaullieu, Jon Crissey, Ian Smith, University of Texas at San Antonio